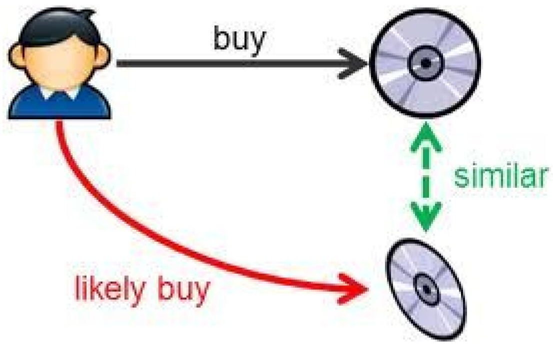
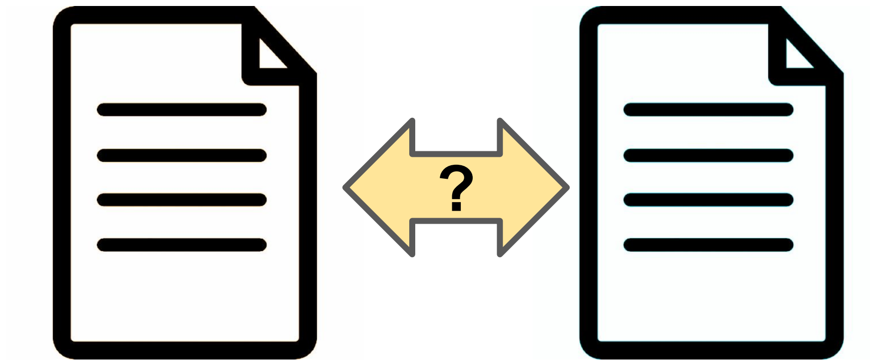
Similarity & Link Analysis

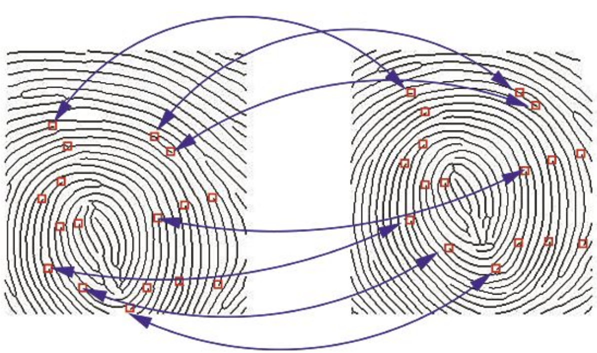
Stony Brook University
CSE545, Fall 2016

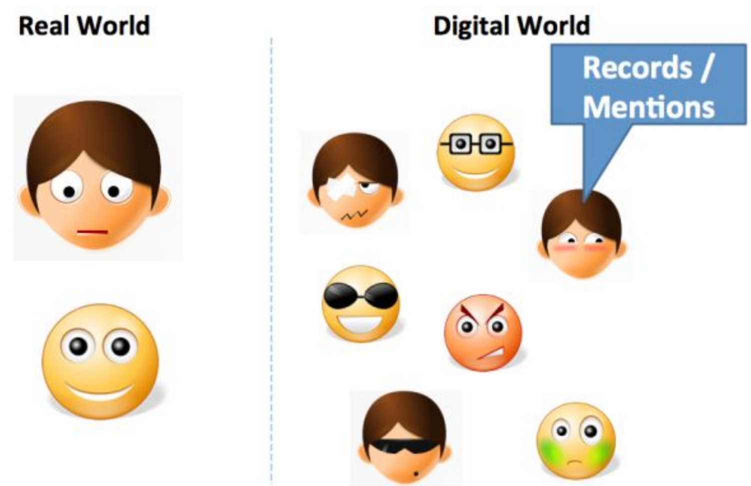
Finding Similar “Items”



(http://blog.soton.ac.uk/hive/2012/05/10/recommendation-system-of-hive/)







(http://www.datacommunitydc.org/blog/2013/08/entity-resolution-for-big-data)

Finding Similar “Items”: What we will cover

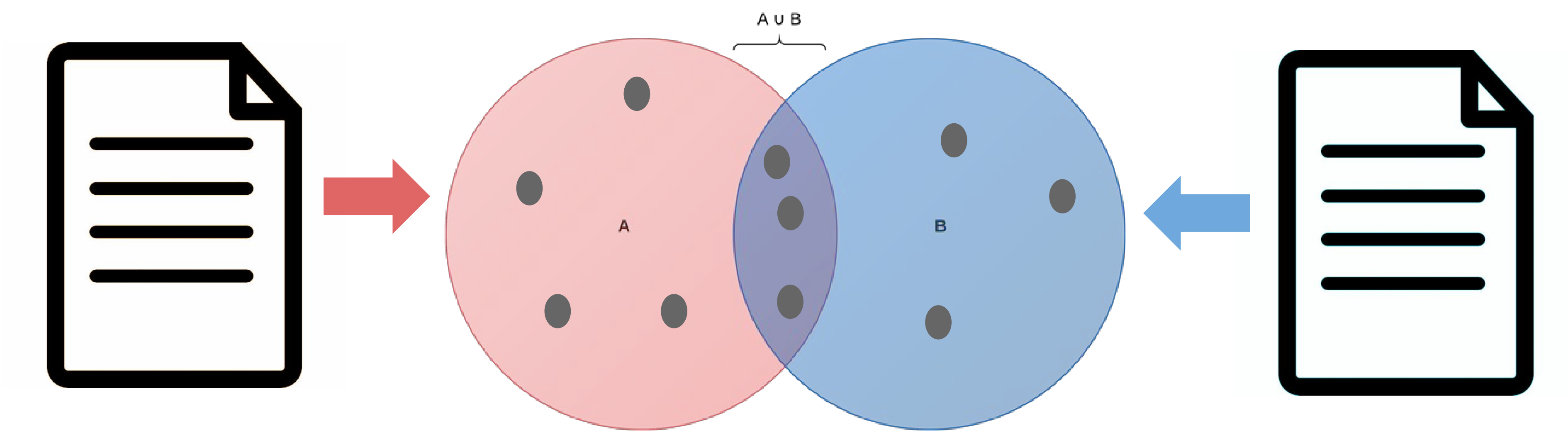
- Shingling
- Minhashing
- Locality-sensitive hashing
- Distance Metrics

Document Similarity

Challenge: How to represent the document in a way that can be efficiently encoded and compared?

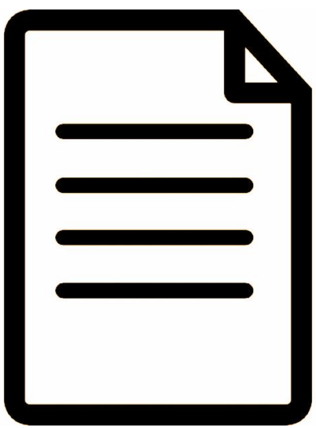
Shingles

Goal: Convert documents to sets



Shingles

Goal: Convert documents to sets

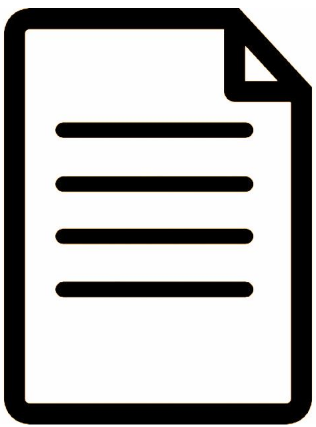


k-shingles (aka “character n-grams”)
- sequence of *k* characters

E.g. k=2 doc=”abcdabd”
singles(doc, 2) = {ab, bc, cd, da, bd}

Shingles

Goal: Convert documents to sets



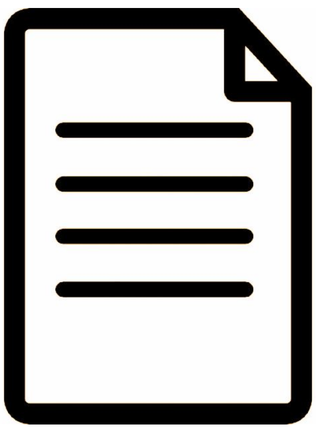
k-shingles (aka “character n-grams”)
- sequence of *k* characters

E.g. *k*=2 doc=”abcdabd”
singles(doc, 2) = {ab, bc, cd, da, bd}

- Similar documents have many common shingles
- Changing words or order has minimal effect.
- In practice use 5 < k < 10

Shingles

Goal: Convert documents to sets



Large enough that any given shingle appearing a document is highly unlikely
(e.g. < .1% chance)

Can hash large shingles to smaller
(e.g. 9-shingles into 4 bytes)

Can also use words (aka n-grams).

s")

acters

a, bd}

- Similar documents have many common shingles
- Changing words or order has minimal effect.
- **In practice use 5 < k < 10**

Shingles

Problem: Even if hashing, sets of shingles are large
(e.g. 4 bytes => 4x the size of the document).

Minhashing

Goal: Convert sets to shorter ids, signatures

Minhashing - Background

Goal: Convert sets to shorter ids, signatures

Characteristic Matrix, X:

$Element$	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

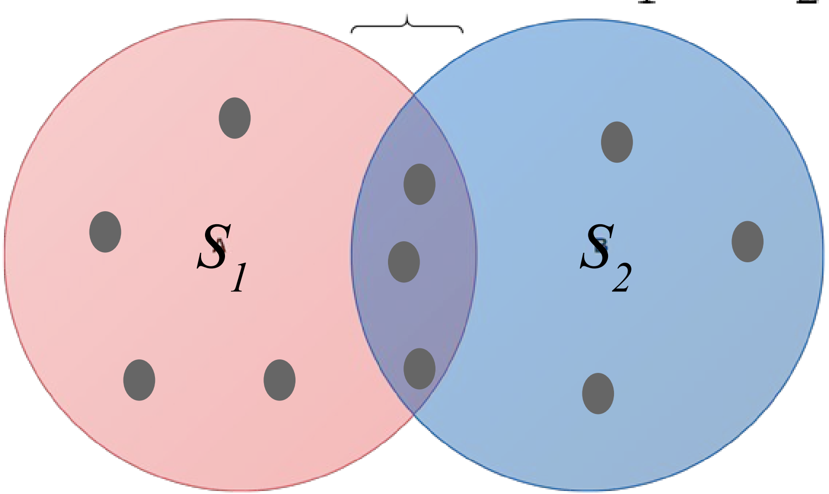
....

(Leskovec at al., 2014; http://www.mmds.org/)

often very sparse! (lots of zeros)

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$



Minhashing - Background

Characteristic Matrix:

	S_1	S_2
ab	1	1
bc	0	1
de	1	0
ah	1	1
ha	0	0
ed	1	1
ca	0	1

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

Minhashing - Background

Characteristic Matrix:

	S_1	S_2	
ab	1	1	* *
bc	0	1	*
de	1	0	*
ah	1	1	**
ha	0	0	
ed	1	1	**
ca	0	1	*

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

Minhashing - Background

Characteristic Matrix:

	S_1	S_2	
ab	1	1	* *
bc	0	1	*
de	1	0	*
ah	1	1	**
ha	0	0	
ed	1	1	**
ca	0	1	*

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

$sim(S_1, S_2) = 3 / 6$

both have / # at least one has

Shingles

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

Minhashing

Characteristic Matrix: X

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Approximate Approach:

1) Instead of keeping whole characteristic matrix, just keep first row where 1 is encountered.

2) Shuffle and repeat to get a "signature" for each set.

Idea: We don't need to actually shuffle we can just use hash functions.

Minhashing

Characteristic Matrix:

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

Minhashing

Characteristic Matrix:

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ =

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ = ed #permuted row 2
$h(S_4)$ =

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ = ed #permuted row 2
$h(S_4)$ = ha #permuted row 1

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1

$h_1(S_1)$ = ed #permuted row 2

$h_1(S_2)$ = ha #permuted row 1

$h_1(S_3)$ = ed #permuted row

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1

$h_1(S_1)$ = ed #permuted row 2

$h_1(S_2)$ = ha #permuted row 1

$h_1(S_3)$ = ed #permuted row

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: *h*

- Based on permutation of rows in the characteristic matrix, *h* maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1

$h_1(S_1)$ = ed #permuted row 2

$h_1(S_2)$ = ha #permuted row 1

$h_1(S_3)$ = ed #permuted row

Minhashing

Characteristic Matrix:

			S_1	S_2	S_3	S_4
4	3	ab	1	0	1	0
2	4	bc	1	0	0	1
1	7	de	0	1	0	1
3	6	ah	0	1	0	1
6	1	ha	0	1	0	1
7	2	ed	1	0	1	0
5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2				

Minhashing

Characteristic Matrix:

			S_1	S_2	S_3	S_4
4	3	ab	1	0	1	0
2	4	bc	1	0	0	1
1	7	de	0	1	0	1
3	6	ah	0	1	0	1
6	1	ha	0	1	0	1
7	2	ed	1	0	1	0
5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Minhashing

Characteristic Matrix: X

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

Minhashing

Characteristic Matrix:

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	[illegible]	0
3	2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
6	3	6	ah	[illegible]	[illegible]	[illegible]	[illegible]
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

Estimate with a random sample of permutations (i.e. ~100)

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	[illegible]	0
3	2	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
6	3	6	ah	[illegible]	[illegible]	[illegible]	[illegible]
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

Estimate with a random sample of permutations (i.e. ~100)

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

Try Sim(S_2, S_4) and Sim(S_1, S_2)

(Leskovec at al., 2014; http://www.mmds.org/)

Minhashing

In Practice

Problem:

- Can’t reasonably do permutations (huge space)
- Can’t randomly grab rows according to an order (random disk seeks = slow!)

Minhashing

In Practice

Problem:

- Can’t reasonably do permutations (huge space)
- Can’t randomly grab rows according to an order (random disk seeks = slow!)

Solution: Use “random” hash functions.

- Setup:
  - Pick ~100 hash functions, hashes
  - Store M[i][s] = a potential minimum $h_i(r)$ *#initialized to infinity (num hashs x num sets)*

Minhashing

Solution: Use "random" hash functions.

- Setup:
  - Pick ~100 hash functions, hashes
  - Store M[i][s] = a potential minimum $h_i(r)$
    #initialized to infinity (num hashs x num sets)
- Algorithm:

```
for r in rows of cm: #cm is characteristic matrix
  compute h_i(r) for all i in hashes #precompute 100 values
  for each set s in row r:
    if cm[r][s] == 1:
      for i in hashes: #check which hash produces smallest value
        if h_i(r) < M[i][s]: M[i][s] = h_i(r)
```

Minhashing

Solution: Use "random" hash functions.

- Setup:
  - Pick ~100 hash functions, hashes
  - Store M[i][s] = a potential minimum $h_i(r)$

    #initialized to i

- Algorithm:

Known as "efficient minhashing".

```
for r in rows of cm: #cm is characteristic matrix
  compute h_i(r) for all i in hashes #precompute 100 values
  for each set s in row r:
    if cm[r][s] == 1:
      for i in hashes: #check which hash produces smallest value
        if h_i(r) < M[i][s]: M[i][s] = h_i(r)
```

Minhashing

What hash functions to use?

Start with 2 decent hash functions

e.g. $h_a(x)$ = ascii(string) % large_prime_number

$h_b(x) = (3*$ascii(string) + $16)$ % large_prime_number

https://www.eecs.harvard.edu/~michaelm/postscripts/rsa2008.pdf

Minhashing

What hash functions to use?

Start with 2 decent hash functions

e.g. $h_a(x)$ = ascii(string) % large_prime_number

$h_b(x) = (3*$ascii(string) $+ 16)$ % large_prime_number

Add together multiplying the second times i:

$h_i(x) = h_a(x) + i*h_b(x)$

e.g. $h_5(x) = h_a(x) + 5*h_b(x)$

https://www.eecs.harvard.edu/~michaelm/postscripts/rsa2008.pdf

Minhashing

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

Minhashing

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

New Problem: Even if the size of signatures are small, it can be computationally expensive to find similar pairs.

E.g. 1m documents; 1,000,000 choose 2 = 500,000,000,000 pairs

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

If we wanted the similarity for all pairs of documents, could anything be done?

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

Approach: Hash multiple times over subsets of data: similar items are likely in the same bucket once.

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

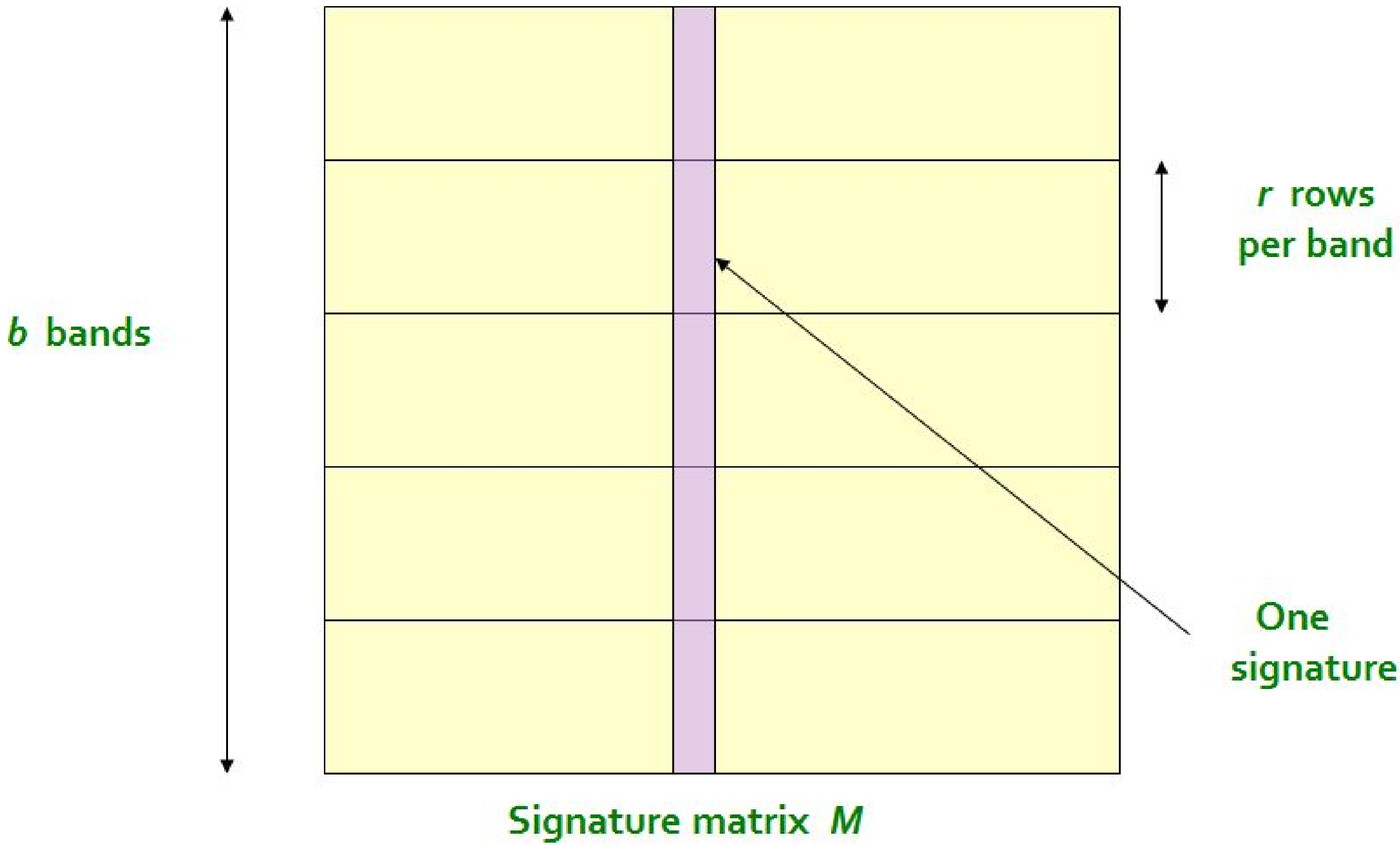
Approach: Hash multiple times over subsets of data: similar items are likely in the same bucket once.

Approach from MinHash: Hash columns of signature matrix

$\Rightarrow$ Candidate pairs end up in the same bucket.

Step 1: Add bands

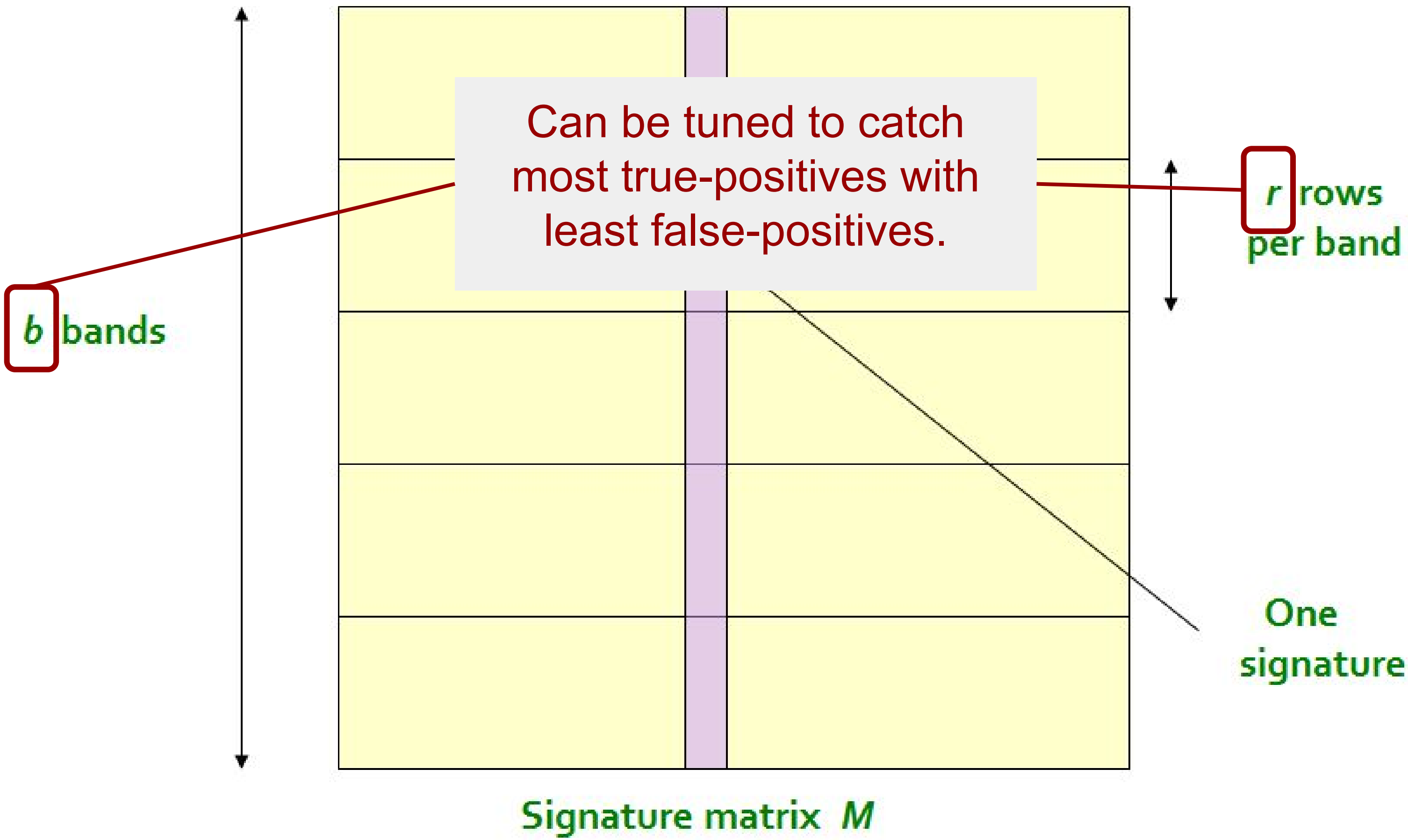
Locality-Sensitive Hashing



(Leskovec at al., 2014; http://www.mmds.org/)

Step 1: Add bands

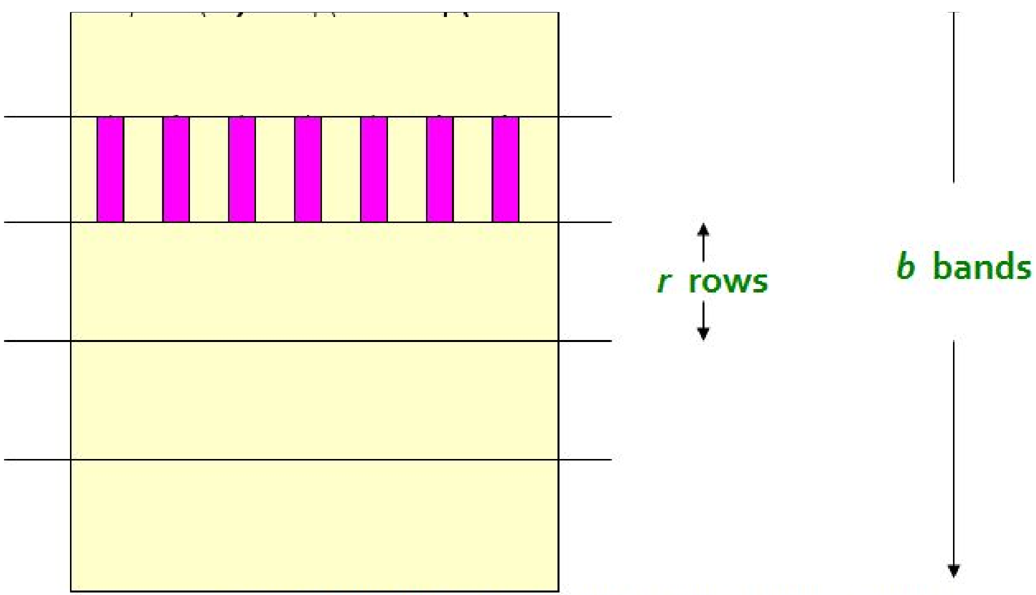
Locality-Sensitive Hashing



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

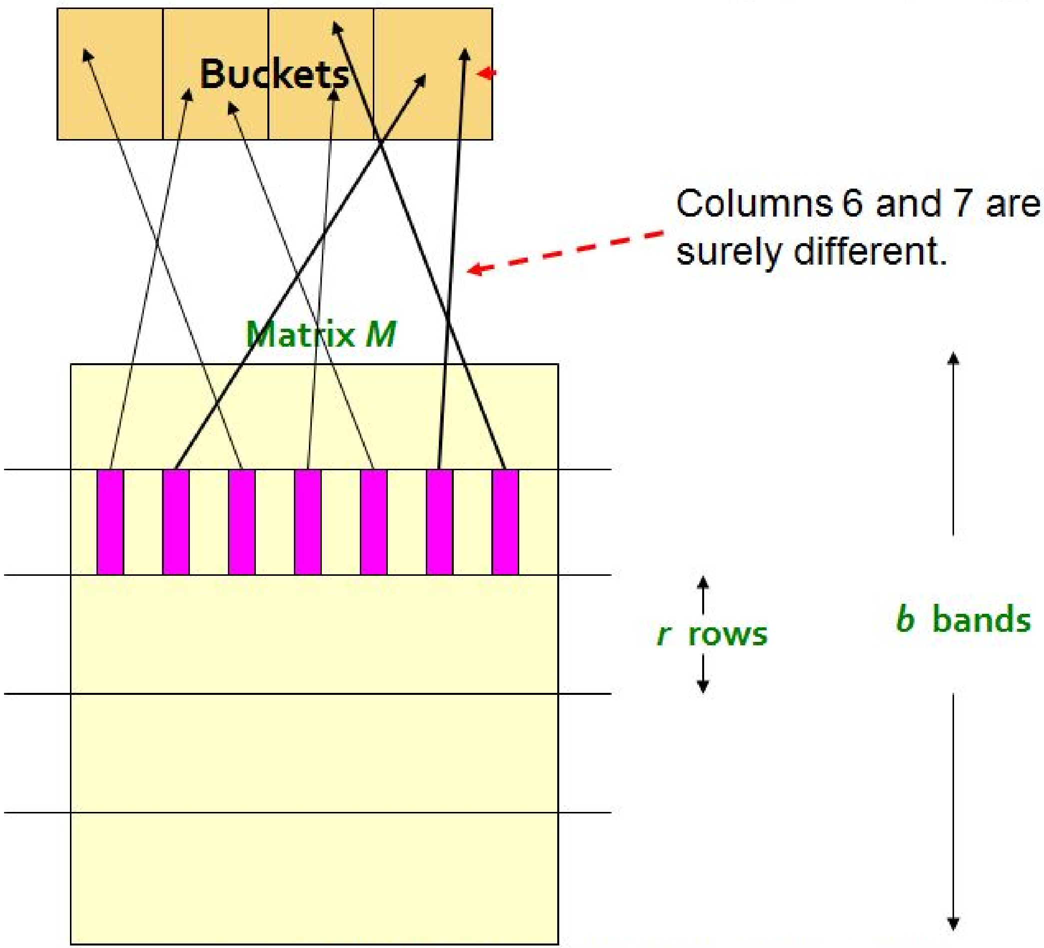
Step 1: Add bands
Step 2: Hash columns within bands



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

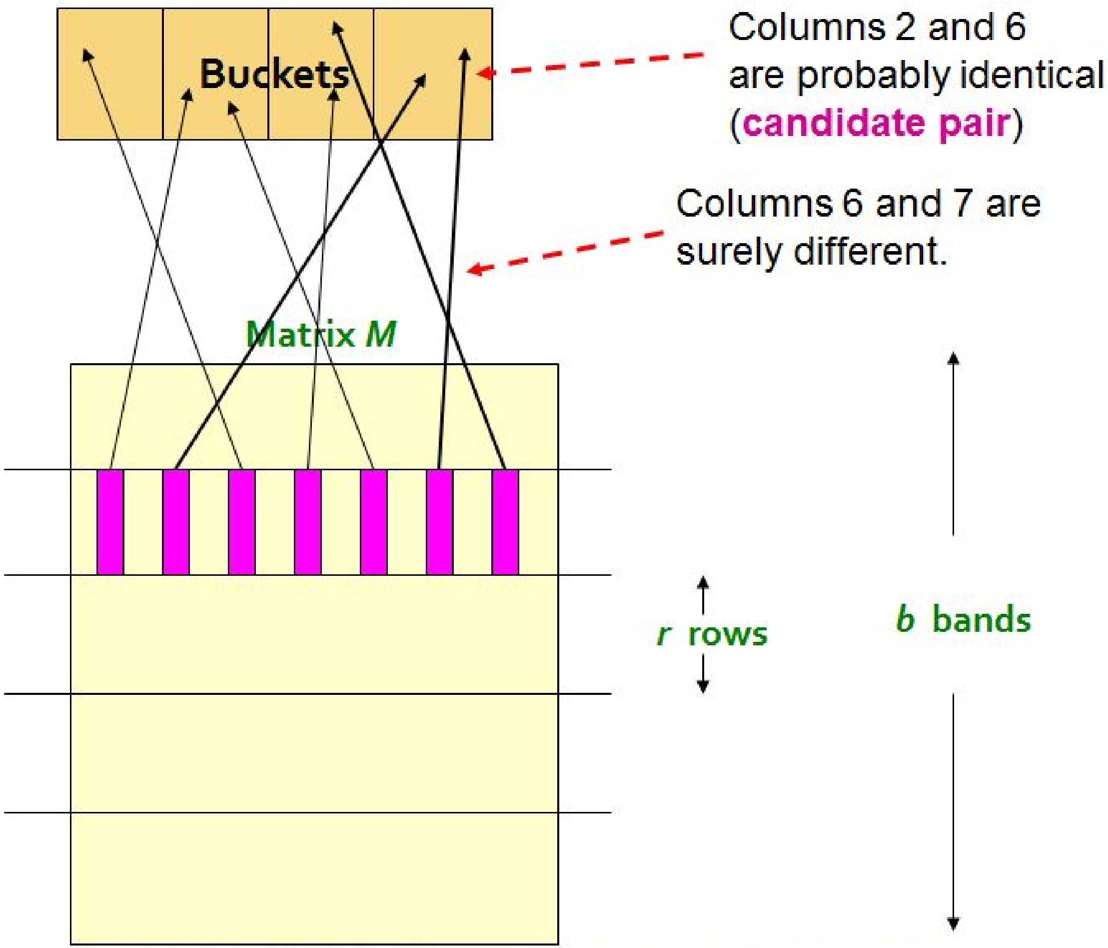
Step 1: Add bands
Step 2: Hash columns within bands



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

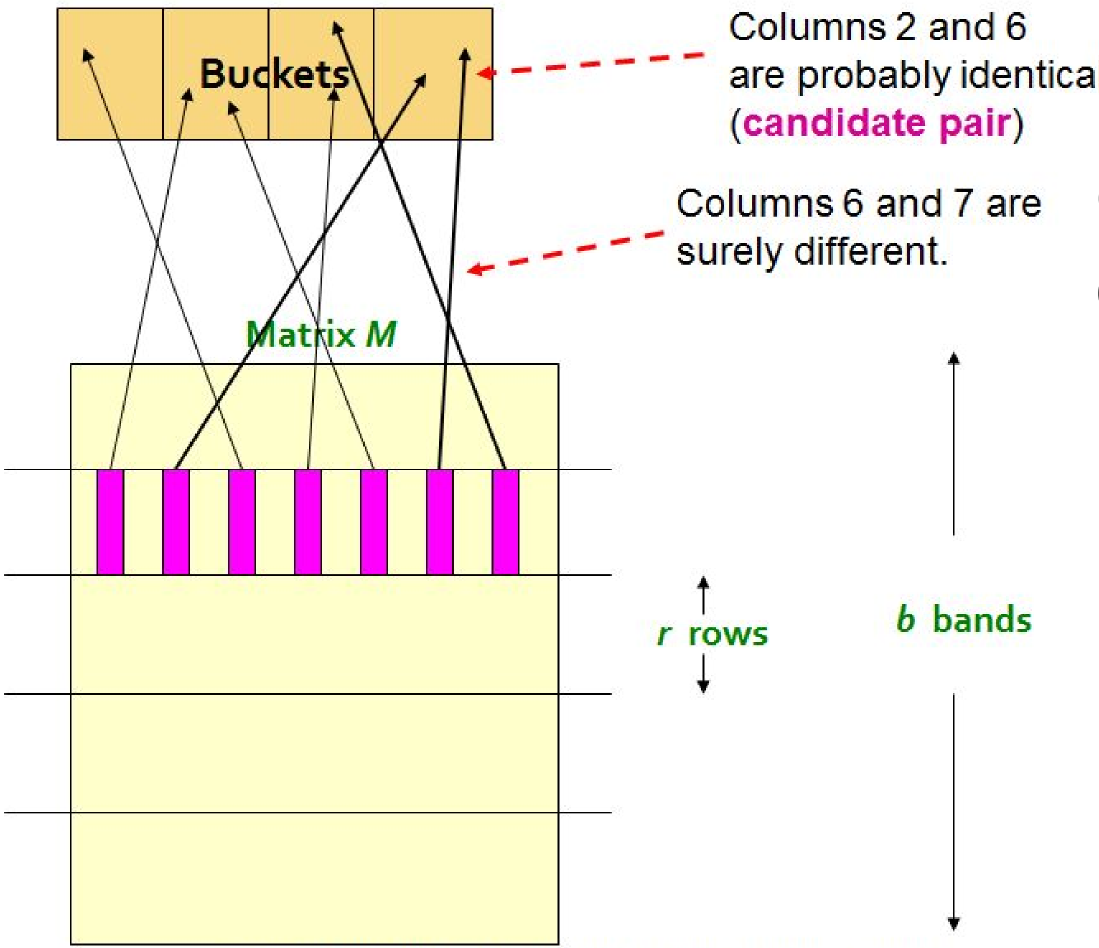
Step 1: Add bands
Step 2: Hash columns within bands



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

Step 1: Add bands
Step 2: Hash columns within bands



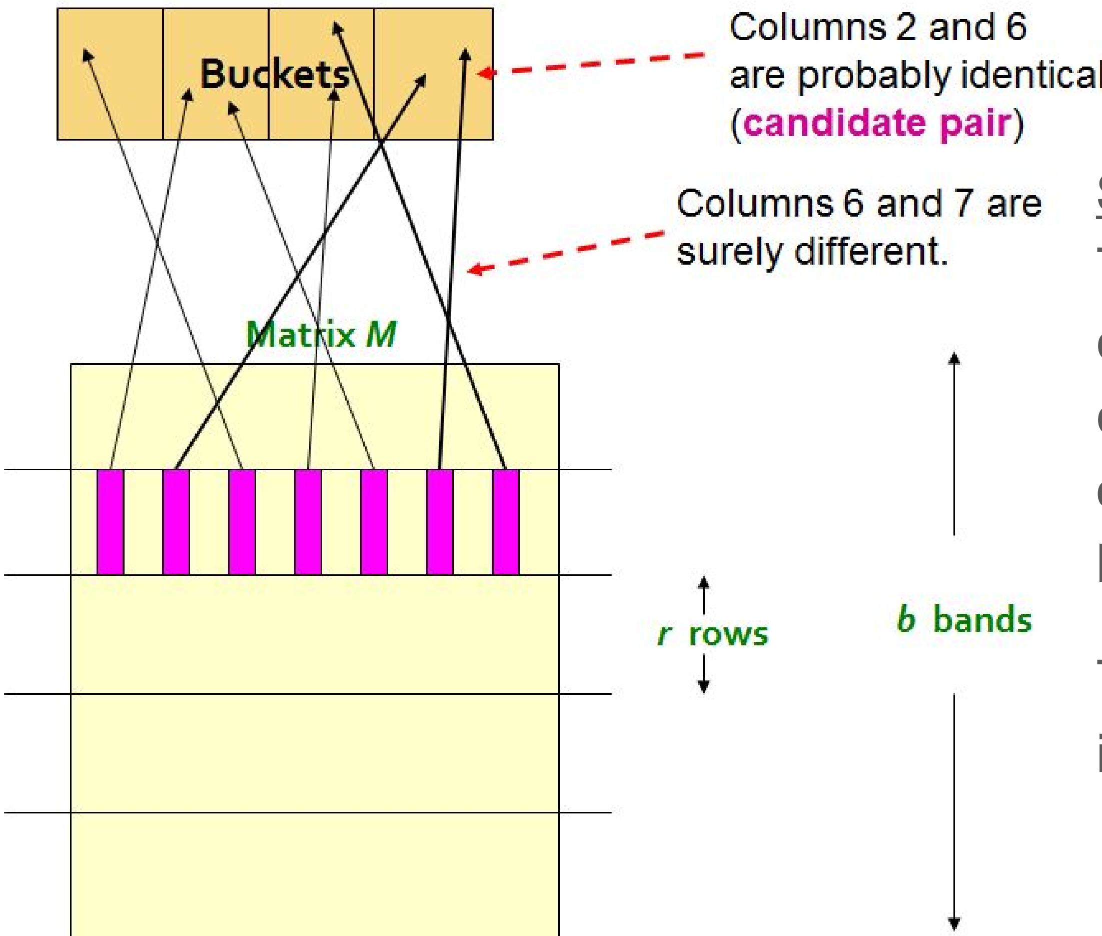
Criteria for being candidate pair:

- They end up in same bucket for at least 1 band.

(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

Step 1: Add bands
Step 2: Hash columns within bands

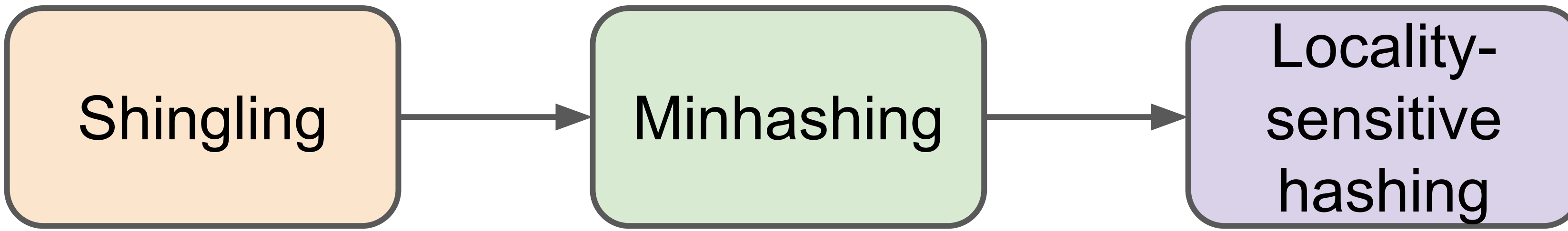


Simplification:
There are enough buckets compared to rows per band that columns must be identical in order to hash to the same bucket.

Thus, we only need to check if identical within a band.

(Leskovec at al., 2014; http://www.mmds.org/)

Document Similarity Pipeline



Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1 == S_2 \mid b)$: probability S1 and S2 agree within a given band

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1{==}S_2 \mid b)$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad \Rightarrow \quad P(S_1 {!=} S_2 \mid b) = 1 - .328 = .672$

$P(S_1 {!=} S_2)$: probability S1 and S2 do not agree in any band

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1{==}S_2 \mid b)$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad \Rightarrow \quad P(S_1 != S_2 \mid b) = 1-.328 = .672$

$P(S_1 != S_2)$: probability S1 and S2 do not agree in any band

$=.672^{20} = .00035$

Realistic Example: Probabilities of agreement

- 100,000 documents
- 100 random permutations/hash functions/rows
  => if 4byte integers then 40Mb to hold signature matrix
  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1 == S_2 \mid b)$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad \Rightarrow \quad P(S_1 != S_2 \mid b) = 1 - .328 = .672$

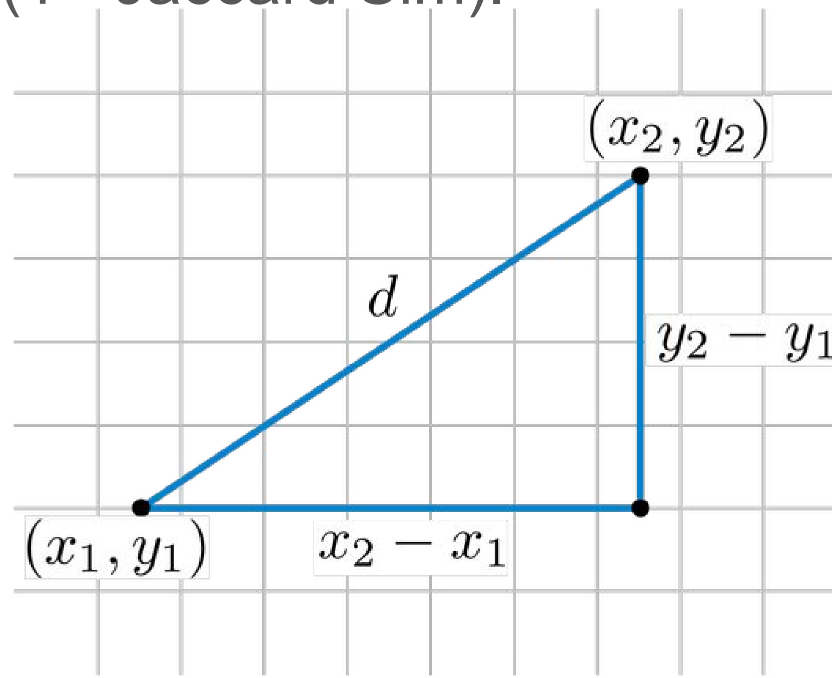
$P(S_1 != S_2)$: probability S1 and S2 do not agree in any band

$= .672^{20} = .00035$

What if wanting 40% Jaccard Similarity?

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).



(http://rosalind.info/glossary/euclidean-distance/)

Distance Metrics

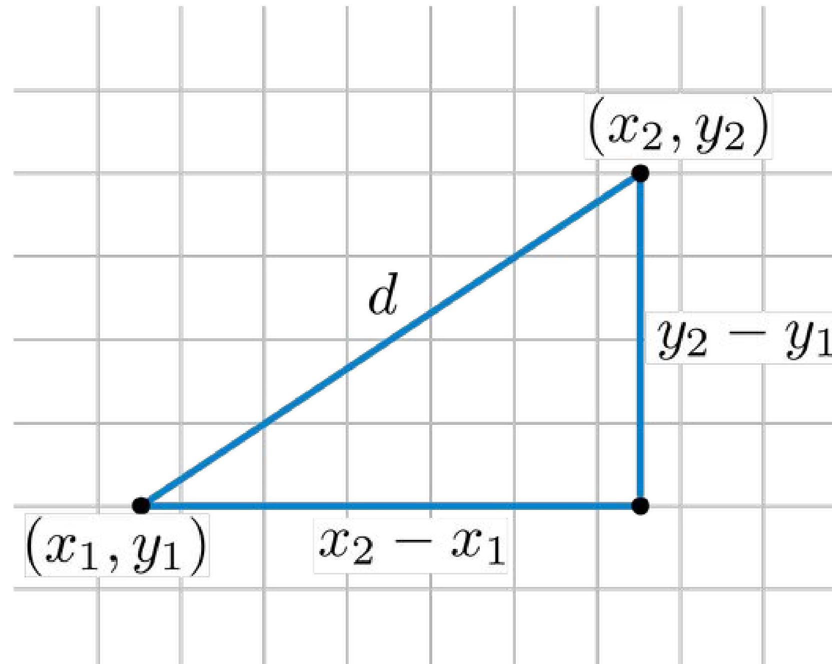
Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

Typical properties of a distance metric, *d*:

d(x, x) = 0

d(x, y) = d(y, x)

d(x, y) ≤ d(x,z) + d(z,y)



(http://rosalind.info/glossary/euclidean-distance/)

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

There are other metrics of similarity. e.g:

- Euclidean Distance
- Cosine Distance

  …
- Edit Distance
- Hamming Distance

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

There are other metrics of similarity. e.g:

- Euclidean Distance

$$distance(X,Y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$ ("L2 Norm")

- Cosine Distance

…

- Edit Distance
- Hamming Distance

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

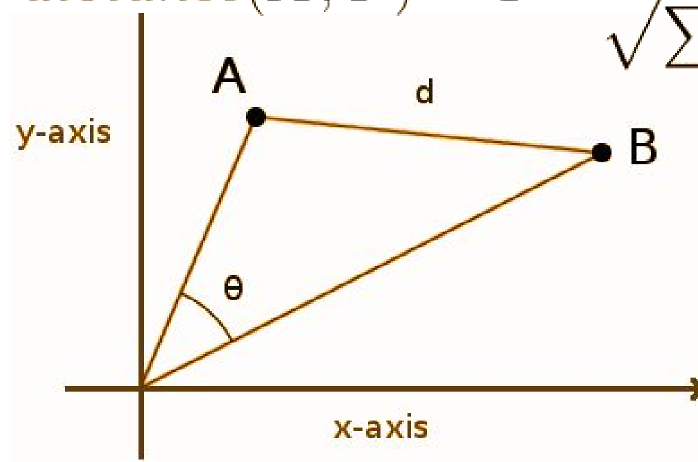
There are other metrics of similarity. e.g:

- Euclidean Distance

$$distance(X,Y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$ ("L2 Norm")

- Cosine Distance

$$distance(X,Y) = 1 - \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$



…

- Edit Distance
- Hamming Distance

Locality Sensitive Hashing - Theory

LSH Can be generalized to many distance metrics by converting output to a probability and providing a lower bound on probability of being similar.

Locality Sensitive Hashing - Theory

LSH Can be generalized to many distance metrics by converting output to a probability and providing a lower bound on probability of being similar.

E.g. for euclidean distance:

- Choose random lines (analogous to hash functions in minhashing)
- Project the two points onto each line; match if two points within an interval

Link Analysis

The Web , circa 1998

The Web , circa 1998

Match keywords, language (*information retrieval*)

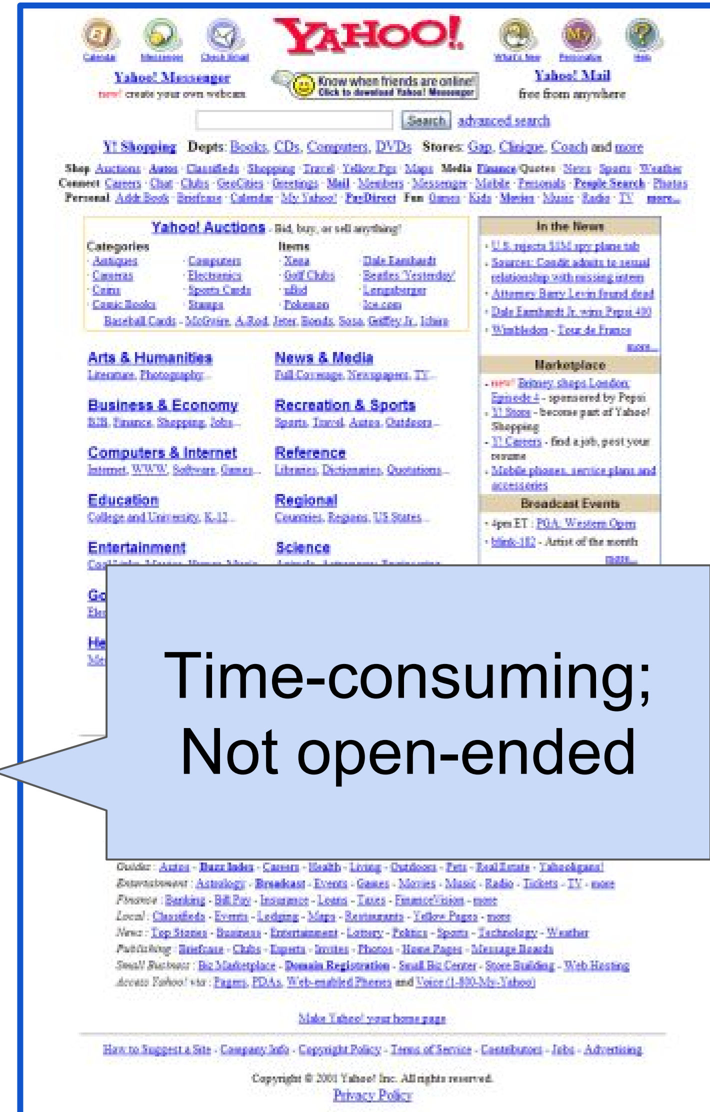
Explore directory

The Web , circa 1998

Easy to game with “term spam”

Match keywords, language (*information retrieval*)

Explore directory



Time-consuming; Not open-ended

Enter PageRank

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy
use of the structure
and produce much
text and hyperlink

...

The PageRank Citation Ranking: Bringing Order to the Web

January 29, 1998

Abstract

The importance of a Web page is an inherently subjective matter, which depends on the readers interests, knowledge and attitudes. But there is still much that can be said objectively

PageRank

Key Idea: Consider the citations of the website.

PageRank

Key Idea: Consider the citations of the website.

Who links to it?
and what are
their citations?

PageRank

Key Idea: Consider the citations of the website.

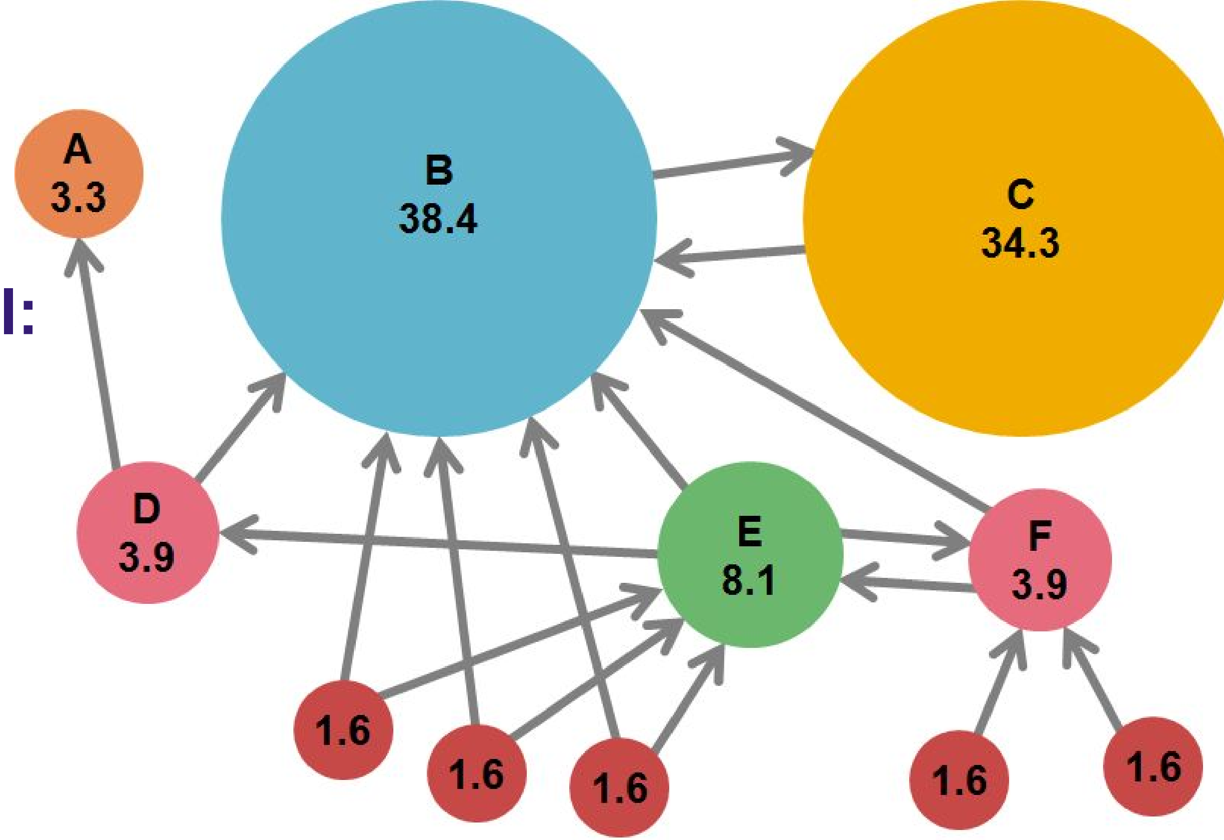
Who links to it?
and what are
their citations?

Innovation 1: What pages would a “random Web surfer” end up at?

Innovation 2: Not just own terms but what terms are used by citations?

PageRank

View 1: Flow Model:
in-links as votes



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Innovation 1: What pages would a "random Web surfer" end up at?

Innovation 2: Not just own terms but what terms are used by citations?

PageRank

View 1: Flow Model:

in-links (citations) as votes

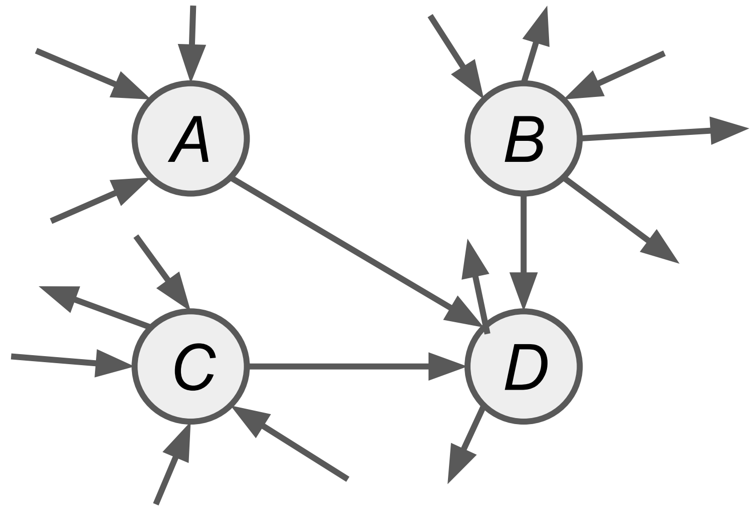
but, citations from important pages should count more.

=> Use recursion to figure out if each page is important.

Innovation 1: What pages would a "random Web surfer" end up at?

Innovation 2: Not just own terms but what terms are used by citations?

PageRank



View 1: Flow Model:

How to compute?

Each page (j) has an importance (i.e. rank, r_j)

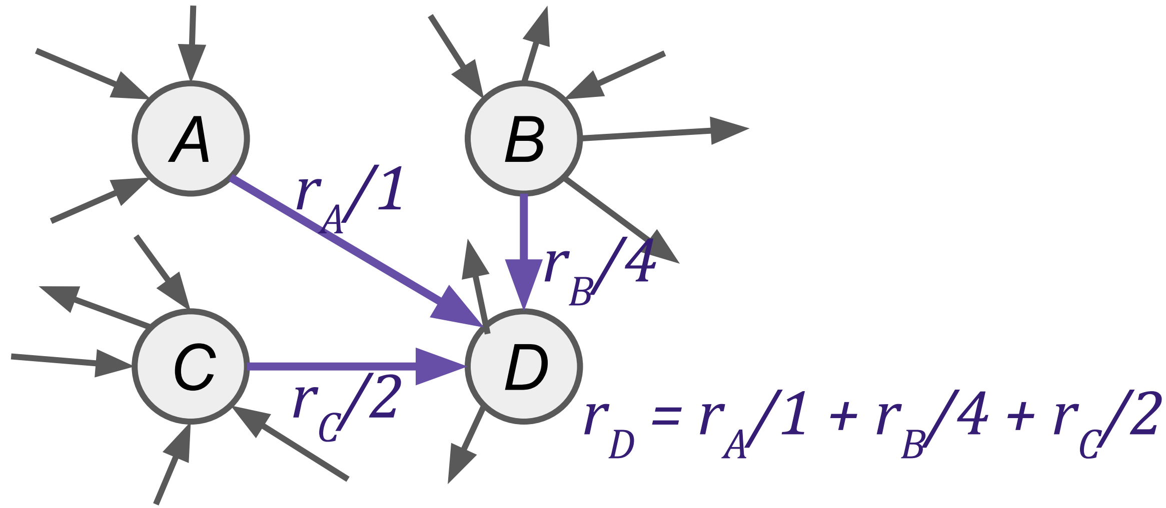
$$vote_j = \frac{r_j}{n_j}$$

(n_j is |out-links|)

$$r_j = \sum_{i \in inLinks(j)} vote_i$$

PageRank

View 1: Flow Model:



How to compute?

Each page (j) has an importance (i.e. rank, r_j)

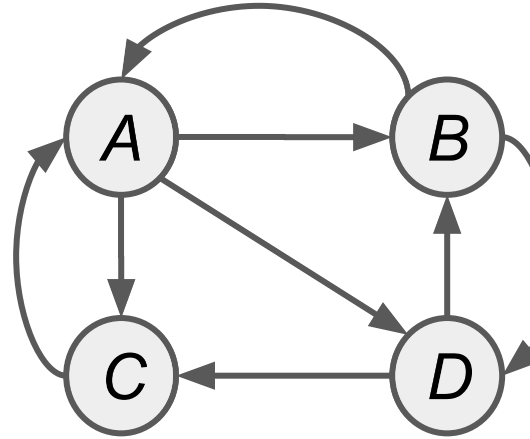
$$vote_j = \frac{r_j}{n_j}$$

(n_j is |out-links|)

$$r_j = \sum_{i \in inLinks(j)} vote_i$$

PageRank

View 1: Flow Model:



How to compute?

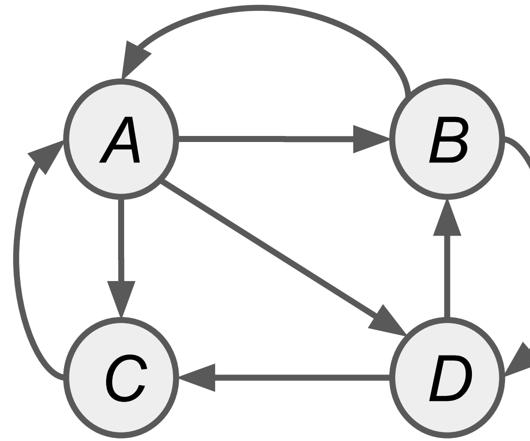
Each page (j) has an importance (i.e. rank, r_j)

$$vote_j = \frac{r_j}{n_j}$$

(n_j is |out-links|)

$$r_j = \sum_{i \in inLinks(j)} vote_i$$

PageRank



View 1: Flow Model:

A System of Equations:

$$r_A = \frac{r_B}{2} + \frac{r_C}{1}$$

How to compute?

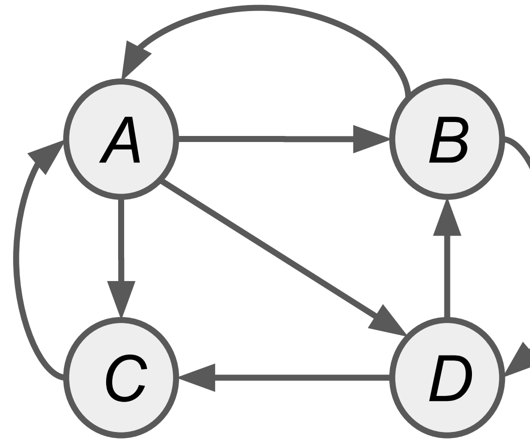
Each page (j) has an importance (i.e. rank, r_j)

$$vote_j = \frac{r_j}{n_j}$$

(n_j is |out-links|)

$$r_j = \sum_{i \in inLinks(j)} vote_i$$

PageRank



View 1: Flow Model:

A System of Equations:

$$r_A = \frac{r_B}{2} + \frac{r_C}{1}$$
$$r_B = \frac{r_A}{3} + \frac{r_D}{2}$$
$$r_C = \frac{r_A}{3} + \frac{r_D}{2}$$
$$r_D = \frac{r_A}{3} + \frac{r_B}{2}$$

How to compute?

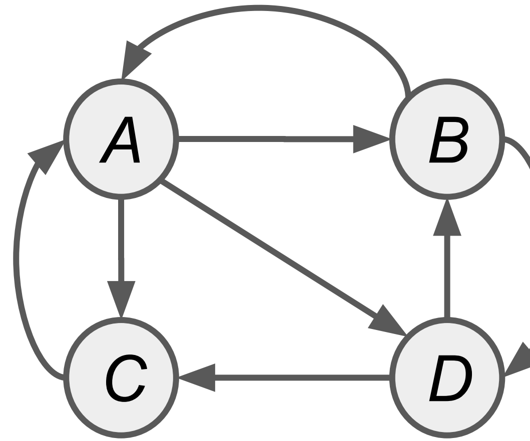
Each page (j) has an importance (i.e. rank, r_j)

$$vote_j = \frac{r_j}{n_j}$$

(n_j is |out-links|)

$$r_j = \sum_{i \in inLinks(j)} vote_i$$

PageRank



View 1: Flow Model: Solve

$$1 = r_A + r_B + r_C + r_D$$

$$r_A = \frac{r_B}{2} + \frac{r_C}{1}$$
$$r_B = \frac{r_A}{3} + \frac{r_D}{2}$$
$$r_C = \frac{r_A}{3} + \frac{r_D}{2}$$
$$r_D = \frac{r_A}{3} + \frac{r_B}{2}$$

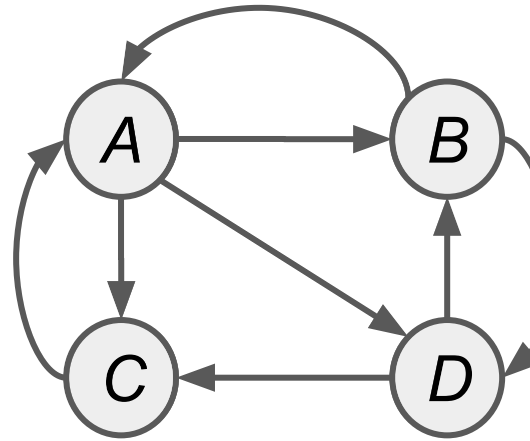
How to compute?

Each page (j) has an importance (i.e. rank, r_j)

$$vote_j = \frac{r_j}{n_j}$$ (n_j is |out-links|)

$$r_j = \sum_{i \in inLinks(j)} vote_i$$

PageRank



$$1 = r_A + r_B + r_C + r_D$$

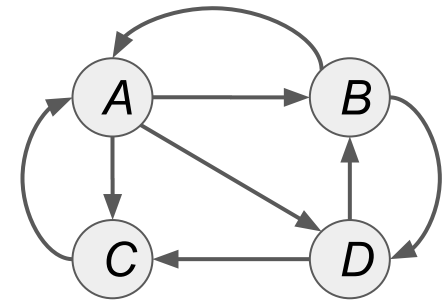
$$r_A = \frac{r_B}{2} + \frac{r_C}{1}$$
$$r_B = \frac{r_A}{3} + \frac{r_D}{2}$$
$$r_C = \frac{r_A}{3} + \frac{r_D}{2}$$
$$r_D = \frac{r_A}{3} + \frac{r_B}{2}$$

to \ from	***A***	***B***	***C***	***D***
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

Transition Matrix, M

Innovation: What pages would a “random Web surfer” end up at?

To start: N=4 nodes, so *r* = [¼, ¼, ¼, ¼,]



View 2: Matrix Formulation

$$1 = r_A + r_B + r_C + r_D$$

$$r_A = \frac{r_B}{2} + \frac{r_C}{1}$$

$$r_B = \frac{r_A}{3} + \frac{r_D}{2}$$

$$r_C = \frac{r_A}{3} + \frac{r_D}{2}$$

$$r_D = \frac{r_A}{3} + \frac{r_B}{2}$$

to \ from	***A***	***B***	***C***	***D***
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

Transition Matrix, M

Innovation: What pages would a "random Web surfer" end up at?

To start: N=4 nodes, so r = [¼, ¼, ¼, ¼,]

after 1st iteration: $M \cdot r$ = [3/8, 5/24, 5/24, 5/24]

after 2nd iteration: $M(M \cdot r) = M^2 \cdot r$ = [15/48, 11/48, …]

View 2: Matrix Formulation

$$1 = r_A + r_B + r_C + r_D$$

$$r_A = \frac{r_B}{2} + \frac{r_C}{1}$$

$$r_B = \frac{r_A}{3} + \frac{r_D}{2}$$

$$r_C = \frac{r_A}{3} + \frac{r_D}{2}$$

$$r_D = \frac{r_A}{3} + \frac{r_B}{2}$$

to \ from	***A***	***B***	***C***	***D***
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

Transition Matrix, M

Innovation: What pages would a “random Web surfer” end up at?

To start: N=4 nodes, so r = [¼, ¼, ¼, ¼,]

after 1st iteration: $M \cdot r$ = [3/8, 5/24, 5/24, 5/24]

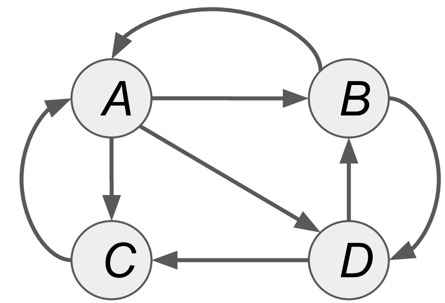
after 2nd iteration: $M(M \cdot r) = M^2 \cdot r = [15/48, 11/48, \ldots]$

Power iteration algorithm

```
initialize:  r[0] = [1/N, …, 1/N],
             r[-1]=[0,...,0]
while (err_norm(r[t],r[t-1])>min_err):



err_norm(v1, v2) = |v1 - v2| #L1 norm
```



to \ from	A	B	C	D
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

“Transition Matrix”, M

Innovation: What pages would a "random Web surfer" end up at?

To start: N=4 nodes, so r = [¼, ¼, ¼, ¼,]

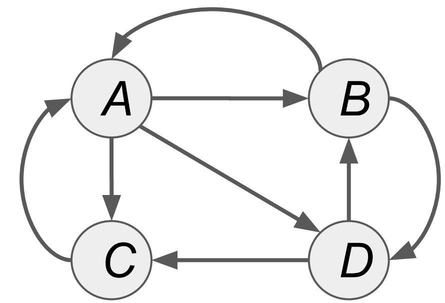
after 1st iteration: $M \cdot r$ = [3/8, 5/24, 5/24, 5/24]

after 2nd iteration: $M(M \cdot r) = M^2 \cdot r$ = [15/48, 11/48, …]

Power iteration algorithm

```
initialize:  r[0] = [1/N, …, 1/N],
             r[-1]=[0,...,0]
while (err_norm(r[t],r[t-1])>min_err):
    r[t+1] = M·r[t]
    t+=1
solution = r[t]

err_norm(v1, v2) = |v1 - v2| #L1 norm
```



to \ from	***A***	***B***	***C***	***D***
A	0	1/2	1	0
B	1/3	0	0	1/2
C	1/3	0	0	1/2
D	1/3	1/2	0	0

"Transition Matrix", *M*

As err_norm gets smaller we are moving toward: $r = M \cdot r$

View 3: Eigenvectors:

Power iteration algorithm

```
initialize:  r[0] = [1/N, …, 1/N],
             r[-1]=[0,...,0]
while (err_norm(r[t],r[t-1])>min_err):
    r[t+1] = M·r[t]
    t+=1
solution = r[t]

err_norm(v1, v2) = |v1 - v2| #L1 norm
```

As err_norm gets smaller we are moving toward: $r = M \cdot r$

View 3: Eigenvectors:

We are actually just finding the ***eigenvector*** of M.

finds the...

Power iteration algorithm

```
initialize:  r[0] = [1/N, …, 1/N]
             r[-1]=[0,...,0]
while (err_norm(r[t],r[t-1])>min_err):
    r[t+1] = M·r[t]
    t+=1
solution = r[t]

err_norm(v1, v2) = |v1 - v2| #L1 norm
```

x is an *eigenvector* of λ if:
A·**x** = λ·**x**

As err_norm gets smaller we are moving toward: $r = M \cdot r$

View 3: Eigenvectors:

We are actually just finding the ***eigenvector*** of M.

finds the...

Power iteration algorithm

```
initialize:  r[0] = [1/N, …, 1/N]
             r[-1]=[0,...,0]
while (err_norm(r[t],r[t-1])>min_err):
   r[t+1] = M·r[t]
   t+=1
solution = r[t]

err_norm(v1, v2) = |v1 - v2| #L1 norm
```

x is an *eigenvector* of λ if:
$A \cdot \mathbf{x} = \lambda \cdot \mathbf{x}$

A = 1
since columns of M sum to 1.
thus, *1r=Mr*

View 4: Markov Process

Where is surfer at time t+1? $p(t+1) = M \cdot p(t)$

Suppose: $p(t+1) = p(t)$, then $p(t)$ is a *stationary distribution* of a ***random walk***.

Thus, r is a stationary distribution. Probability of being at given node.

View 4: Markov Process

Where is surfer at time t+1? $p(t+1) = M \cdot p(t)$

Suppose: $p(t+1) = p(t)$, then $p(t)$ is a *stationary distribution* of a ***random walk***.

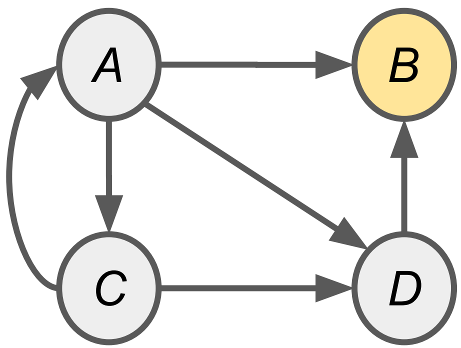
Thus, r is a stationary distribution. Probability of being at given node.

aka 1st order Markov Process

- Rich probabilistic theory. One finding:
  - Stationary distributions have a unique distribution if:
    - No *"dead-ends"*: a node can't propagate its rank
    - No *"spider traps"*: set of nodes with no way out.

Also known as being *stochastic*, *irreducible*, and *aperiodic.*

View 4: Markov Process - Problems for vanilla PI



to \ from	***A***	***B***	***C***	***D***
A	0	0	1	0
B	1/3	0	0	1
C	1/3	0	0	0
D	1/3	0	0	0

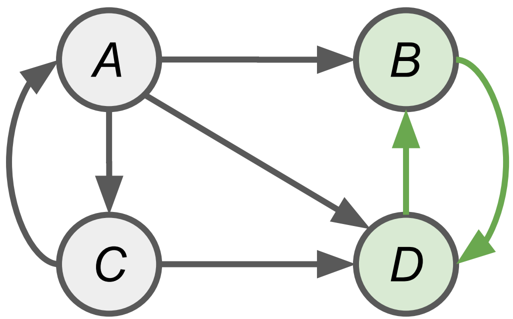
What would *r* converge to?

aka 1st order Markov Process

- Rich probabilistic theory. One finding:
  - Stationary distributions have a unique distribution if:
    - No “***dead-ends***”: a node can’t propagate its rank
    - No “*spider traps*”: set of nodes with no way out.

Also known as being *stochastic*, *irreducible*, and *aperiodic.*

View 4: Markov Process - Problems for vanilla PI



to \ from	***A***	***B***	***C***	***D***
A	0	0	1	0
B	1/3	0	0	1
C	1/3	0	0	0
D	1/3	1	0	0

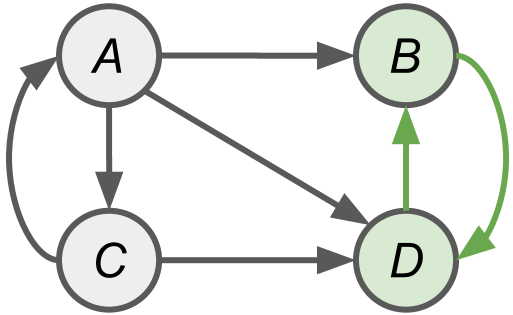
What would *r* converge to?

aka 1st order Markov Process

- Rich probabilistic theory. One finding:
  - Stationary distributions have a unique distribution if:
    - No *"dead-ends"*: a node can't propagate its rank
    - No ***"spider traps"***: set of nodes with no way out.

Also known as being *stochastic*, *irreducible*, and *aperiodic.*

View 4: Markov Process - Problems for vanilla PI



to \ from	**A**	**B**	**C**	**D**
A	0	0	1	0
B	1/3	0	0	1
C	1/3	0	0	0
D	1/3	1	0	0

What would *r* converge to?

aka 1st order Markov Process

- Rich probabilistic theory. One finding:
  - Stationary distributions have a unique distribution if:

same node doesn't repeat at regular intervals

columns sum to 1 non-zero chance of going to any other node

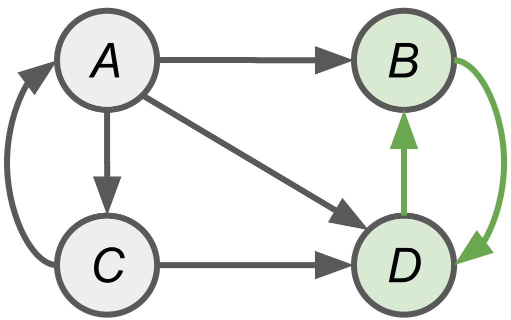
Also known as being *stochastic*, *irreducible*, and *aperiodic*.

Goals:
No “dead-ends”
No “spider traps”

The “Google” PageRank Formulation

Add teleportation:At each step, two choices

1. Follow a random link (probability, β = ~.85)
2. Teleport to a random node (probability, 1-β)

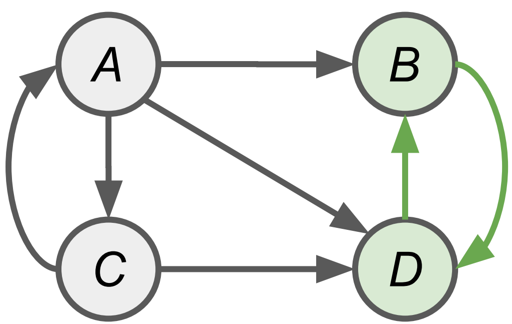


Goals:
No "dead-ends"
No "spider traps"

The "Google" PageRank Formulation

Add teleportation:At each step, two choices

1. Follow a random link (probability, β = ~.85)
2. Teleport to a random node (probability, 1-β)



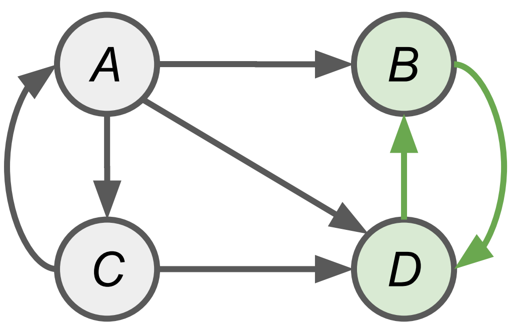
to \ from	***A***	***B***	***C***	***D***
A	0	0	1	0
B	⅓	0	0	1
C	⅓	0	0	0
D	⅓	1	0	0

Goals:
No “dead-ends”
No “spider traps”

The “Google” PageRank Formulation

Add teleportation:At each step, two choices

1. Follow a random link (probability, β = ~.85)
2. Teleport to a random node (probability, 1-β)



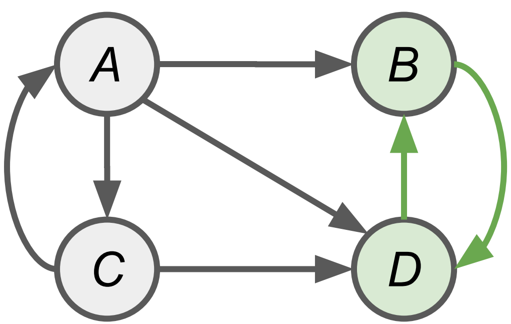
to \ from	***A***	***B***	***C***	***D***
A	0	0+.15*¼	1	0+.15*¼
B	⅓	0+.15*¼	0	.85*1+.15*¼
C	⅓	0+.15*¼	0	0+.15*¼
D	⅓	.85*1 +.15*¼	0	0+.15*¼

Goals:
No “dead-ends”
No “spider traps”

The “Google” PageRank Formulation

Add teleportation:At each step, two choices

1. Follow a random link (probability, β = ~.85)
2. Teleport to a random node (probability, 1-β)



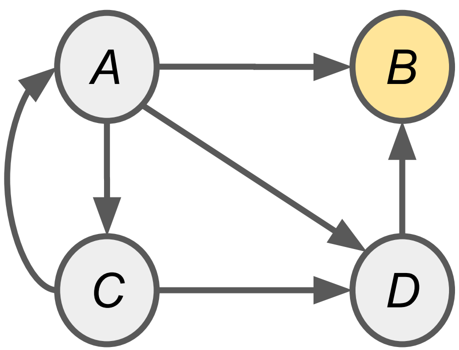
to \ from	***A***	***B***	***C***	***D***
A	0+.15*¼	0+.15*¼	85*1+.15*¼	0+.15*¼
B	.85*⅓+.15*¼	0+.15*¼	0+.15*¼	.85*1+.15*¼
C	.85*⅓+.15*¼	0+.15*¼	0+.15*¼	0+.15*¼
D	.85*⅓+.15*¼	.85*1+.15*¼	0+.15*¼	0+.15*¼

Goals:
No "dead-ends"
No "spider traps"

The "Google" PageRank Formulation

Add teleportation:At each step, two choices

1. Follow a random link (probability, β = ~.85)
2. Teleport to a random node (probability, 1-β)



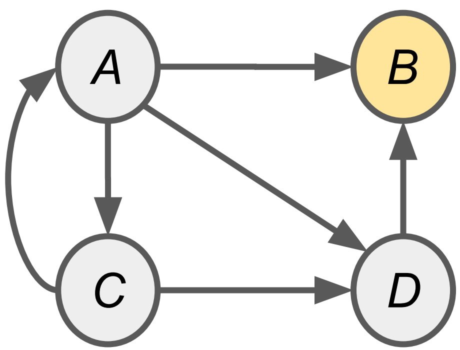
to \ from	***A***	***B***	***C***	***D***
A	0	0	1	0
B	⅓	0	0	1
C	⅓	0	0	0
D	⅓	0	0	0

Goals:
No "dead-ends"
No "spider traps"

The "Google" PageRank Formulation

Add teleportation:At each step, two choices

1. Follow a random link (probability, β = ~.85)
2. Teleport to a random node (probability, 1-β)



to \ from	***A***	***B***	***C***	***D***
A	0	¼	1	0
B	⅓	¼	0	1
C	⅓	¼	0	0
D	⅓	¼	0	0

Goals:

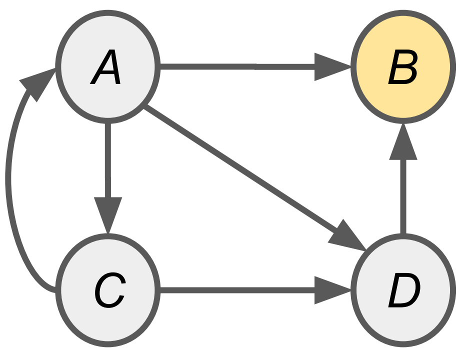
No "dead-ends"

No "spider traps"

The "Google" PageRank Formulation

Add teleportation:At each step, two choices

1. Follow a random link (probability, β = ~.85)
2. Teleport to a random node (probability, 1-β)



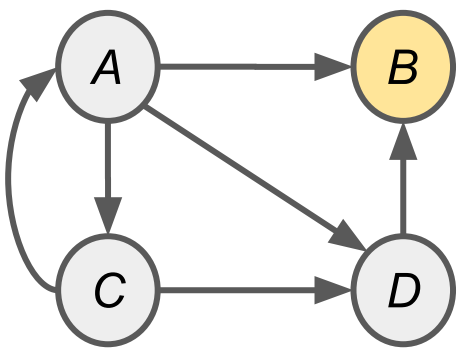
to \ from	***A***	***B***	***C***	***D***
A	0	.85*¼+.15*¼	1	0
B	⅓	.85*¼+.15*¼	0	1
C	⅓	.85*¼+.15*¼	0	0
D	⅓	.85*¼+.15*¼	0	0

Goals:
No “dead-ends”
No “spider traps”

The “Google” PageRank Formulation

Add teleportation:At each step, two choices

1. Follow a random link (probability, β = ~.85)
2. Teleport to a random node (probability, 1-β)
   (Teleport from a dead-end has probability 1)



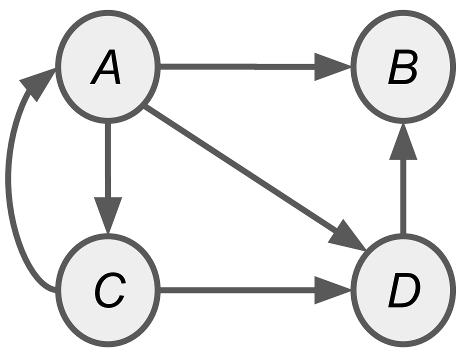
to \ from	***A***	***B***	***C***	***D***
A	0+.15*¼	1*¼	85*1+.15*¼	0+.15*¼
B	.85*⅓+.15*¼	1*¼	0+.15*¼	.85*1+.15*¼
C	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼
D	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼

Goals:
No "dead-ends"
No "spider traps"

Teleportation, as Flow Model:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

(Brin and Page, 1998)



to \ from	***A***	***B***	***C***	***D***
A	0+.15*¼	1*¼	85*1+.15*¼	0+.15*¼
B	.85*⅓+.15*¼	1*¼	0+.15*¼	.85*1+.15*¼
C	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼
D	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼

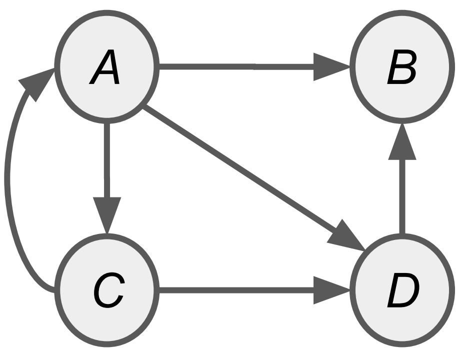
Goals:
No "dead-ends"
No "spider traps"

Teleportation, as Flow Model:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

(Brin and Page, 1998)

Teleportation, as Matrix Model: $M' = \beta M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$



to \ from	***A***	***B***	***C***	***D***
A	0+.15*¼	1*¼	85*1+.15*¼	0+.15*¼
B	.85*⅓+.15*¼	1*¼	0+.15*¼	.85*1+.15*¼
C	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼
D	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼

Goals:
No "dead-ends"
No "spider traps"

Teleportation, as Flow Model:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

(Brin and Page, 1998)

Teleportation, as Matrix Model:

$$M' = \beta M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$$

to \ from	***A***	***B***	***C***	***D***
A	0+.15*¼	.85*¼+.15*¼	85*1+.15*¼	0+.15*¼
B	.85*⅓+.15*¼	.85*¼+.15*¼	0+.15*¼	.85*1+.15*¼
C	.85*⅓+.15*¼	.85*¼+.15*¼	0+.15*¼	0+.15*¼
D	.85*⅓+.15*¼	.85*¼+.15*¼	0+.15*¼	0+.15*¼

Goals:
No "dead-ends"
No "spider traps"

Teleportation, as Flow Model:

$$r_j = \sum_{i \to j} \beta \, \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

(Brin and Page, 1998)

Teleportation, as Matrix Model:

$$M' = \beta \, M + (1-\beta)\left[\frac{1}{N}\right]_{N \times N}$$

To apply:
run power iterations over M' instead of M.

to \ from	***A***	***B***	***C***	***D***
A	0+.15*¼	1*¼	85*1+.15*¼	0+.15*¼
B	.85*⅓+.15*¼	1*¼	0+.15*¼	.85*1+.15*¼
C	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼
D	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼

Goals:
No “dead-ends”
No “spider traps”

Teleportation, as Flow Model:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1-\beta)\frac{1}{N}$$

(Brin and Page, 1998)

Teleportation, as Matrix Model: $M' = \beta\, M + (1-\beta)\left[\frac{1}{N}\right]_{N\times N}$

Steps:
1. Compute M
2. Add 1/N to all dead-ends.
3. Convert *M* to *M’*
4. Run Power Iterations.

to \ from	***A***	***B***	***C***	***D***
A	0+.15*¼	1*¼	85*1+.15*¼	0+.15*¼
B	.85*⅓+.15*¼	1*¼	0+.15*¼	.85*1+.15*¼
C	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼
D	.85*⅓+.15*¼	1*¼	0+.15*¼	0+.15*¼